In a Telco-CDN, Pushing Content Makes Sense

Zhe Li, Gwendal Simon

In a Telco-CDN, Pushing Content Makes Sense

Zhe Li and Gwendal Simon

***Abstract*—The exploding HD video streaming traffic calls for deploying content servers deeper inside network operators' infrastructures. Telco-CDN are new content distribution services that are managed by Internet Service Providers (ISP). Since the network operator controls both the infrastructure and the content delivery overlay, it is in a position to engineer telco-CDN so that networking resources are optimally utilized. In this paper, we show the following two findings: 1. it is possible to implement an efficient algorithm for the placement of video chunks into a telco-CDN. We present an algorithm, which is based on a genetic algorithm implemented on the MapReduce framework. We show that, for a national VoD service, computing a quasi-optimal placement is possible. 2. such push strategy makes sense because it allows to actually take into account fine-grain traffic management strategies on the underlying infrastructure.**

Our proposal re-opens the debate about the relevance of such "push" approach (where the manager of telco-CDN proactively pushes video content into servers) versus the traditional caching approach (where the content is pulled to the servers from requests of clients). Our proposal of a quasi-optimal tracker enables fair comparisons between both approaches for most traffic engineering policies. We illustrate the interest of our proposal in the context of a major European Telco-CDN with real traces from a popular Video-on-Demand (VoD) service. Our experimental results show that, given a perfect algorithm for predicting user preferences, our placement algorithm is able to keep aligned with LRU caching in terms of the traditional hit-ratio, but the workload on some troubled links (*e.g.,* over-used links) in a push-based strategy is significantly alleviated.

***Index Terms*—CDN, ISP, Optimal content placement, In-network caching;**

I. Introduction

Major Internet Service Providers (ISPs) are considering the deployment of a Content Delivery Network (CDN) in their own network infrastructure. Such deployment is not trivial for network operators, which do traditionally not operate repository servers and data-centers, but it has become an evidence from a business perspective since the Internet is turning into a content-centric infrastructure. Thus, the management of a so-called *telco-CDN* is today a critical business concern as well as a scientific problem of growing importance [19, 27, 36, 37]. In Section III, we will develop today's CDN market analysis and explain in details the rationale behind the proliferation of telco-CDNs.

From a scientific standpoint, the management of CDN has been studied for more than a decade [34]. However, the characteristics of telco-CDN differ from those of a traditional CDN and thus challenge some of the certitudes everybody agreed on. In particular, the goal of a traditional CDN is to optimize the performance of an *overlay*, subject to some constraints related to an underlying network infrastructure. On the contrary, a telco-CDN is under the responsibility of a network operator that owns both the overlay and the underlying infrastructure. The main performance metric is fundamentally different because the network operator is concerned with the management of its underlying network. A network operator typically manages the traffic so that it reduces the load on links that experience congestion and it favors traffic on links being under-utilized. The joint management of overlay and underlay in the context of CDN is a critical challenge for network operators.

In this paper, we re-open a well-known debate [16, 24, 34, 44, 45], which is central in the management of a CDN in general, and is critical in the specific case of a telco-CDN. *Should the operator decide the placement of content into its servers, or should it implement a caching strategy for its servers?* This latter strategy has been adopted by most CDNs so far [32] because it is very simple to implement, and near optimal in terms of cache hit, which is the main performance metric in traditional CDN [41]. Every server dynamically replaces the content it stores based on a caching policy that takes into account latest requests from clients. The *Least Recently Used* (LRU) caching policy is known to be especially efficient despite its gorgeous simplicity [23]. In comparison, the "push" strategy, where the operator decides the location of content into every server, is harder to implement for a negligible benefit. The push strategy requires indeed an efficient algorithm that predicts the future requests from end-users. Moreover, determining the best placement is an optimization problem in the family of Facility Location Problems (FLP), which are commonly NP-complete.

We re-open this debate for three reasons. Firstly, as previously said, the objective of a telco-CDN is different. The impact of the CDN on the infrastructure becomes a major metric. The goal is to minimize what we call the *infrastructure cost*, which is a metric that is given by the network operator and reflects the performances of the traffic management policy on Telco-CDN underlying links. Secondly, the performances of recommendation and prediction algorithms have significantly grown during the past years [47]. These progresses make large-scale content producer consider again prefetching video [24]. Thirdly, the size of the overlay is smaller than what was commonly considered in previous studies, which targeted worldwide CDN services. For a typical Telco-CDN in a national ISP, the number of deployed repositories is less than hundred [39], which cannot be compared to the 100,000 servers of Akamai [17]. Thus, it becomes possible to implement sophisticated algorithms.

This paper makes two major contributions:

We present a practical, efficient algorithm for content placement in telco-CDN. The optimal video placement is modeled by the k-Product Capacitated Facility Location Prob-

Zhe Li and Gwendal Simon works in the Network Department of Telecom Bretagne (Institut Mines-Telecom), France

lem (k-PCFLP). Our idea here is to leverage a well-known *meta-heuristic*, namely genetic algorithm (GA), to reach a good level of performances. We say that our algorithm is efficient in the sense that literature related to genetic algorithm consistently shows close-to-optimal performances. In addition, we present an implementation of this GA on the *MapReduce* framework, which enables computation of large telco-CDN instances on a small cluster of machines. Content placement is done on a regular basis from the observations from ISP (network status and video popularity).

We present in a realistic evaluation the benefits one can expect from a push strategy for telco-CDN. Our algorithm enables the comparison with traditional caching strategies. We collected real traces from a VoD service that should typically be hosted in a telco-CDN. The data set consists of more than $700{,}000$ requests from more than $20{,}000$ end-users distributed over 13 geographical areas. We study a telco-CDN deployment that is currently investigated by a major European network operator (Orange) and a simple but enlightening traffic management policy. Our main observation is that LRU caching performs as well as our push strategy for the hit ratio. However, a push strategy enables an accurate management of the underlying infrastructure, so it is a solution that is worth the implementation for network operators.

II. Related Work

We argued in Section I that previous research works on CDN are not suitable for designing Telco-CDN since they consider only CDN overlay and their cache hit-ratio [33]. Our contribution addresses the problem of underlay management in CDN. While most commercial CDN providers choose LRU caching in practice, our results show that cooperative caching is not smart enough for an ISP that should consider both underlay and overlay.

The placement of content in ISP networks has been recently investigated with theoretical perspectives [10, 19, 37]. The interactions between ISP and content providers are considered in [19] and [10] with two cases: either content providers adjust their content placement based on changes in the underlying network, or ISP implements content-aware routing to improve their service. These studies do not address the case where ISPs leverage content placement to engineer traffic. The study presented in [37] is the closest work to ours. The main difference is that authors do not consider the preference of ISPs on their internal links. That is, the "cost" for transmitting one unit of data on each internal link is the same. We argue here that real-world network engineering imposes differentiating links with a generic cost function. While the authors in [37] show that simple LRU caching still outperforms sophisticated push-based placement strategies, we show on the contrary that totally different results can be obtained if such a real network management strategy is introduced.

Many replication and content distribution techniques have been studied for P2P system during the last decade. Uniform, proportional and square root replications are analyzed in [7, 13, 29, 38]. Schemes that combine proportional replication and pull based caching have also been presented in [21, 42, 45]. These replication techniques aim at finding a good trade-off between redundancy and cost efficiency of resources. A major issue in P2P networks relates to dynamics of peers. Redundancy aims at dealing with peer churn, but excessive replication may waste peer resources. However, in our context, all the repositories are under the control of an ISP, thus peer churn is not considered as a critical problem. Moreover, most of the replication schemes for P2P systems do not consider engineering underlying network traffic using P2P overlay. The only works related to traffic engineering are the P4P proposal [46] and the subsequent ALTO works. These works aim to reduce inter-domain traffic based on the hints offered by ISPs. But these contributions can hardly apply to fine-grain intra-domain traffic management.

From an algorithmic point of view, FLP is an extensively studied problem in the domain of operational research. Various algorithms are proposed to solve the seminal problem and its variants [18, 40]. However, few of them considers the k-PCFLP variant, especially our case where there are multiple constraints on the service capability of facility. Moreover, these centralized algorithms are not efficient enough to handle large instances such as telco-CDN. MapReduce has been proposed to accelerate the process of some classes of algorithms (*e.g.,* graph algorithm [25]). Particularly, in [20] the authors propose a general framework for parallelizing genetic algorithms using MR. But they do not provide specific solution for any concrete optimization problem. The authors of [9] leverages MR and GA to significantly improve the efficiency of solving the Job Shop Scheduling Problems (JPPS). However, to the best of our knowledge, this is the first work that leverage MR and GA to solve k-PCFLP.

III. Introducing Telco-CDN

Several models have been proposed to capture the reality of relationships between business entities in the sector of content delivery [8, 11, 30]. None of them is fully convincing, as demonstrated by the fact that, under these models, some business entities are engaged in agreements that should theoretically result in economical losses [28]. Moreover, some entities play several roles. For example, in the model proposed by Ma *et al.* [30], Telefonica is all together an Eyeball ISP (an ISP that serve residential users), a Content ISP (an ISP that has deployed a broad infrastructure for content delivery), a Transit ISP and a Content Provider. We are not interested in precisely modeling monetary interactions. We rather aim to understand motivations behind current strategies developed by stakeholders. We roughly distinguish four main entities:

Content providers want to serve their subscribers (residential end-users) without having to engage specific deals with ISPs (excluding potential exclusivity or syndication deals, which are particular use cases). They also want to maintain exclusive relationships with their clients. Since personalization is considered as a promising way to monetize services, the providers want to be notified of each and every action of their clients. Therefore *traffic interception by un-authorized third-parties is not tolerable*. Lastly, service providers wish to drastically lower their cost structure.

CDN providers have deployed a content delivery infrastructure (servers, possibly backbone networks) and have agreements with content providers. They want to remain in the game by returning to more healthy levels of profitability, by getting rid of variable costs and by finding added value in content handling. CDN providers also want to participate to global infrastructure investments, in order to better balance and value their core assets (technologies, knowledge and current infrastructures).

ISPs receive money from residential end-users. They want to maintain cost structures under control in order to remain competitive on the Internet access market (*i.e.* maintaining low pricing/level of margin). They also want to control the global QoS offered to certain services and to maintain a certain level of differentiation.

End-users want to access any service, at any time and from anywhere in the world. As most services are either free or cheap, end-users tend to confound the actors in the value chain, which leads to requests for one overall monthly bill in order to access any service on any device through any network.

The sudden shift in this precarious market equilibrium comes from the explosion of multimedia traffic. Since CDN providers have deployed most of their servers out of ISP networks, bottlenecks starts appearing in the peering links between ISP and CDN networks, which has led to a series of clashes between well-established actors [1, 14]. To overcome this problem, CDN providers would like to deploy more servers directly in ISP networks and to engineer the traffic between residential end-users and these "in-network servers". This is unacceptable for ISPs because CDN providers have no global knowledge on the underlying network.

A. Rationales behind telco-CDNs

The above analysis highlights that many market players objectives converge (in spite of some disparities). The global interest of the value chain players is converging toward a better collaboration. Multiple analysts consider that ISPs should provide new storage spaces with guaranteed links to the customers [35]. This forms the so-called telco-CDN. Some pioneers, including Verizon [43] and Comcast [12], have already built their telco-CDN. As far as we know, other European network operators are also building data-centers. Many operators are also deploying co-location servers with content producers [2].

The raise of telco-CDNs is not necessarily dangerous for traditional CDN providers. The role of the CDNs is to get and distribute contents from the content providers either by its own servers or by delegating this distribution to the telco-CDNs. Current CDN providers are exploring multiple ways to interact with Telco-CDNs. The most intuitive one is to share their advanced experience of content delivery with ISPs [3]. Another way is to coordinate multiple localized Telco-CDNs and to thus assist the creation of an open "federation" of telco-CDNs [4]. The relationships between a CDN and Telco-CDN is out of the scope of this paper.

A collaboration between these actors preserves the essence of the Internet, which is to let every network operator manage its own telco-CDN, according to the characteristics of its network. For instance, an ISP currently investing in a new line of distributed data-centers will leverage these small set of massive storage centers, whereas an ISP presenting a large base of customers equipped with storage-enabled home gateways will focus on mechanisms that exploit these resources. The aggregation of telco-CDNs cooperating with traditional CDNs aligns with domain specific policies.

B. Sketching telco-CDN architecture

We consider for simplicity a single ISP entity, which provides Internet access to end-users and controls the access network, from peering points to the last mile. The idea behind telco-CDN is to install **repositories** near switches and routers. More probably, ISPs leverage the recent deployment of data-centers within their networks. Home gateways and set-top-boxes can also become content servers, as suggested in [45].

We call **tracker** the entity that manages the telco-CDN and interfaces with the external CDNs. These latter upload new contents in the telco-CDN, and delegate their distribution to the end-users that are clients from the ISP.

A typical topology of a telco-CDN is given in Figure 1. We do not represent tracker here. The traditional CDN interfaces with telco-CDN on **Point-of-Presence (PoP)**. There are three major PoPs in France. For this implementation, which is the most probable according to stakeholders, an ISP deploys small data-centers near the routers that connect the backbone core network and the metropolitan access networks of every region (the blue, circular nodes). Thus each telco-CDN repository is in charge of a regional area.

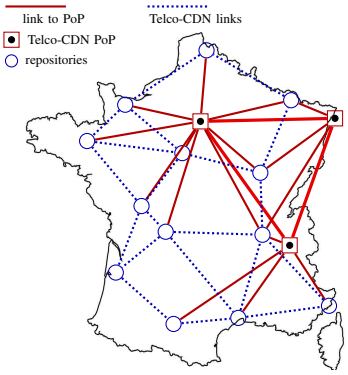


Fig. 1. Envisioned telco-CDN topology in France. Three telco-CDN PoPs enable inter-connections with multiple traditional CDNs.

This scenario is respectful of all actors (video provider, CDN provider and ISP) and is deployable according to the agenda of each actor. It takes advantage of a better network resource utilization, and, contrarily to proposals based on traffic interception [6], it does not bypass the service provider and the CDNs: they still receive the requests from clients, so they are able to monitor and personalize their services.

IV. Optimal Content Placement

We introduce now the problem of finding an optimal content placement with respect to both overlay and underlay in a telco-

CDN. Please refer to Table I for a summary of the notations we will use throughout the paper.

We consider that one "global CDN" orchestrates the distribution of a set of videos K ($l = |K|$) through multiple telco-CDNs. We focus on *one* of these telco-CDNs but our study is generic to any telco-CDN. The network operator that manages this telco-CDN should allocate the videos in K to a set of repositories J ($m = |J|$). In case of a *miss* (when a request for a video in K cannot be fulfilled by the telco-CDN), the requested video can be retrieved from the global CDN but this would be associated with monetary compensations. The set of end-users is noted I ($n = |I|$).

The objective of a network operator is at least twofold. First, it has to preserve the quality of its network infrastructure because it is its core business. Second, a network operator that deploys a telco-CDN should do its best to avoid misses. In order to accommodate both objectives, we introduce two critical parameters that have to be set by the network operator according to its network management strategies.

- **The assignment cost** a_{jk} of pushing a given content $k \in K$ in a given repository $j \in J$. When a content is already in the repository, this cost is null. The main difficulty is to deal with the wide range of possible repositories. The cost of pushing a new content in a residential box is typically bigger than in a data-center, where it is close to null. Please note that assignment cost is frequently neglected with regards to the importance of other costs.
- **The service cost** e_{ij} of fetching a content from the repository $j \in J$ to a client $i \in I$. The setting of this matrix is under the responsibility of the network operator. It is part of the intelligence of network operators to carefully adjust the cost of every link with regard to the quality of the connection and the amount of traffic this link should carry. A very simple way to set service costs is to consider a price that is proportional to the distance between both end-machines.

Our main idea is to integrate the *miss* in the problem formulation by overweighting the service cost of all links toward the PoPs. That is, when no repository stores a given content, a client has to utilize a PoP link in order to fetch data from the global CDN. By overweighting links to the PoPs, we penalize the overall service cost. Therefore a network operator should find a trade-off between the infrastructure cost and the telco-CDN benefit by adjusting the weight of the different service costs. This idea allows the definition of a joint overlay and underlay optimization problem.

The binary variable p_{ik} is the output of the predictions produced either by content providers or by CDNs based on the statistic of user behavior. The variable p_{ik} indicates whether it is highly probable that end-user $i \in I$ will request video $k \in K$ soon. In fact, for a given end-user i, a small subset of videos $K' \subset K$ verifies $p_{ik} = 1, \forall k \in K'$. These videos are typically the dozen of videos recommended by the video service for this user i.

The problem addresses both the placement of video into repositories and the assignment of end-users to repositories.

K, l	set of videos, and size of video catalog
J, n	set of repositories, and number of repositories
I, n	set of end-users and size of the population
a_{jk}	assignment cost for repository j to retrieve video k from CDN.
e_{ij}	service cost for end-user i to obtain a video from repository j.
p_{ik}	recommendation that end-user i requests video k.
s_j	storage capacity of repository j.
b_j	maximum number of end-users that can be allocated to repository j.
x_{jk}	binary variable indicating whether video k is stored on repository j.
y_{ijk}	binary variable indicating whether end-user i obtain video k from repository j.

TABLE I
NOTATIONS

We have two binary variables:

$$x_{jk} = \begin{cases} 1 & \text{if video } k \in K \text{ is stored on repository } j \in J, \\ 0 & \text{otherwise.} \end{cases}$$

which is the placement variable. Then, the redirection of end-users to repositories is captured by:

$$y_{ijk} = \begin{cases} 1 & \text{if end-user } i \text{ obtain video } k \text{ from repository } j, \\ 0 & \text{otherwise.} \end{cases}$$

where a request from user $i \in I$ for a video $k \in K$ should be redirected to server $j \in J$ when $y_{ijk} = 1$.

The storage capacity of each repository $j \in J$ is restricted by s_j video. With the recent development of rate-adaptive video streaming, the size of a video is now bigger than 20 Gbits because a server must store multiple representations of the same movie. We consider a typical value b_j to express that a given repository $j \in J$ cannot be associated to more than b_j clients, which is a common dimensioning constraint even for services at the time scale of a day.

We formulate our k-PCFLP as follows:

$$\text{Minimize} \sum_J \sum_K a_{jk} x_{jk} + \sum_I \sum_J \sum_K e_{ij} p_{ik} y_{ijk}$$

$$\text{subject to} \sum_J y_{ijk} = p_{ik}, \forall i \in I, \forall k \in K \quad (1)$$

$$x_{jk} \geq p_{ik} \cdot y_{ijk}, \forall i \in I, \forall j \in J, \forall k \in K \quad (2)$$

$$\sum_K x_{jk} \leq s_j, \forall j \in J \quad (3)$$

$$\sum_I \sum_K p_{ik} \cdot y_{ijk} \leq b_j, \forall j \in J \quad (4)$$

$$x_{jk} \in \{0,1\}, \forall j \in J, \forall k \in K$$

$$y_{ijk} \in \{0,1\}, \forall i \in I, \forall j \in J, \forall k \in K$$

With constraint (1), the telco-CDN must satisfy each end-user request. In the meantime, every recommended video request is treated by only one repository. Constraint (2) specifies that a repository can provide a video only if the video is placed on it. Finally, constraints (3) and (4) guarantee that the load on a repository does not exceed its service capabilities.

This problem is a k-Product Capacitated Facility Location Problem (k-PCFLP). It is NP-complete [26]. To our knowledge, neither efficient heuristic nor approximate algorithms have been studied for this variant of the FLP family.

V. Efficient tracker

The number of repositories can be relatively small, but the number of videos in a typical VoD service, as well as the number of clients to serve, make the computation of an optimal solution to the problem described in Section IV impossible in practice. Our motivation is to design an algorithm that is efficient and that can be reasonably implemented by ISP for the management of their telco-CDN. We use the term "efficient" to refer to *meta-heuristics*, which have shown in the past that they achieve optimality for most of the instances of a given problem. To improve further the processing time of meta-heuristics, some previous works have explored the deployment of meta-heuristics into massively parallel cloud architectures. On our side, we have implemented a meta-heuristic, namely Genetic Algorithm (GA), on the MapReduce framework. We have chosen GA because we expected GA to be both highly parallelizable and efficient for our problem. In the following, we will first recall the main idea behind GA, then we will present how we model our problem in GA, finally we will describe the MapReduce implementation of this GA.

A. Background on Genetic Algorithms

A Genetic Algorithm mimics the events of the process of natural evolution. It consists of the following steps:

- *Encoding method* transforms a solution into its genetic representation called *individual*, which is usually a series of digital numbers.
- *Fitness function* evaluates the quality of each individual so that the best solutions survive the competition.
- *Selection*, *crossover* and *mutation* are three operations that allow to combine several valid individuals and to produce one so-called *offspring* from them.

The typical procedure of a GA starts with a randomly generated population of v individuals. Next, the algorithm computes the fitness value of each individual in the generation. Then, the three operators are repeated to produce v offspring to evolve the current generation into a new one. The production of generation executes iteratively until some convergence condition is reached. For more information about GA, please refer to [31].

B. Genetic Algorithm for Content Placement

The general idea of our GA is as follows. We generate individuals that correspond to a placement of videos to the repositories. Then we use a fitness function that computes an optimal assignment from end-users to repositories, subject to the constraints of k-PCFLP. Finally, we leverage the inner operations of the genetic algorithm to converge toward a better solution by modifying the best proposed solutions.

1) Encoding method: We create individuals that correspond to a given placement of videos to the repositories, with respect to the storage constraint. In other words, an individual corresponds to a set of $x_{jk}, \forall j \in J, \forall k \in K$. Our goal is to use a representation that enables easy implementation of the operations (selection, crossover, mutation). We chose a $(\sum_{j \in J} s_j)$-uple of integers, which represents the identifiers of the video on the different "storage slots" of repositories. Simply put, this encoding is a list of all $x_{jk}, \forall j \in J, \forall k \in K$ verifying $x_{jk} = 1$, which is sorted in ascending order of j's identifier.

For example, given an instance with $m = 2$ repositories, where the storage capacity of server j_1 (respectively j_2) is $s_{j_1} = 3$ (respectively $s_{j_2} = 2$). Let us assume an individual $A = (a_1, \ldots, a_5) = (1, 3, 4, 2, 5)$. The three first entries a_1, a_2 and a_3 corresponds to the three videos in K that are allocated to repository j_1. Here repository j_1 stores videos k_1, k_3 and k_4. It also implies that $x_{j_1 k} = 0$ for all k in $K \setminus \{k_1, k_3, k_4\}$. The two last entries a_4, and a_5 are the videos allocated to repository j_2 (here k_2 and k_5).

Lemma 1 *Any individual respecting the proposed encoding obeys constraints 3 of the k-PCFLP.*

Proof. Since the overall length of an individual is exactly the sum of all storage capacities, there exists a m-decomposition of an individual where the length of each subset equals the storage capacities of each repository. □

2) Fitness function: The fitness fonction takes in input an individual F and should decide the value of each $y_{ijk}, \forall i \in I, \forall j \in J, \forall k \in K$. If the individual yields a possible solution (every recommendation of video for end-users is assigned to a repository storing this video), the fitness function outputs a final score, which enables competition among individuals.

The fitness function aims at assigning recommendations to repositories. It is another optimization problem related to a matching problem. We propose here a polynomial-time algorithm that computes the optimal solution to this matching problem. We transform the original problem into a *Minimum Cost Maximum Flow* (MCMF) problem as follows. Please refer to Figure 2 for an illustration.

We build a graph containing six classes of vertices. Two of them enable flow computation: (i) a virtual source and (ii) a virtual sinks. We also have the main targets of our computation (iii) end-user nodes and (iv) repositories. Then we add two vertices, which ensure that an end-user is linked to a repository only if this repository stores a video that is recommended to the said end-user: (v) user-video and (vi) repo-video. There exists a user-video vertex noted u_{ik} if and only if $p_{ik} = 1$. Similarly, there exists a repo-video vertex noted r_{jk} if and only if $x_{jk} = 1$ in the individual A that is evaluated by this fitness function.

Links in the graph should primarily make sure that assignment can only be done between an end-user and a repository "sharing" one video (the repository stores a video recommended to the user). We stipulate that a user-video u_{ik} has a link to a repo-video $r_{jk'}$ if and only if $k = k'$. Thus there is a path between the end-user i and the repository j.

We introduce then the flow on the edges. The main idea is that we force each end-user to fetch its video from only one repository, and we force each repository $j \in J$ to not serve more than b_j end-users. To achieve such result, the flow between end-user vertices and user-video vertices are restricted to one unit and the flow between repo-video and repositories vertices are limited by the bandwidth of the repositories.

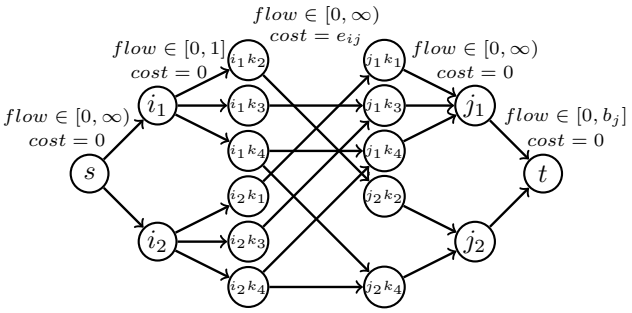


Fig. 2. Example of MCMF

Finally, the cost is set on the links between user-video and repo-video vertices. On a link between a user-video u_{ik} and a repo-video r_{jk}, the cost is set to e_{ij}, for any $k \in K$.

In Figure 2, end-user i_1 is recommended videos in $\{k_2, k_3, k_4\}$, and i_2 videos in $\{k_1, k_3, k_4\}$. The videos k_1 and k_3 are stored on server j_1, while k_2 is offered by j_2. Finally video k_4 is stored by both j_1 and j_2.

The fitness function is an algorithm in two rounds. First, we compute the maximum achievable flow $f(A)$ in the graph built from the individual F. We have:

$$f(A) = \begin{cases} \sum_{i \in I} \sum_{k \in K} p_{ik} & \Rightarrow \text{individual } A \text{ is feasible} \\ \text{otherwise} & \Rightarrow \text{individual } A \text{ is not feasible.} \end{cases}$$

If A is feasible, then we determine the solution that has the minimum cost overall. The fitness function returns this final cost as an output. Otherwise, the fitness function output ∞. Various methods can be used to obtain the optimal solution of MCMF. We implemented the push-relabel method proposed in [5].

Lemma 2 *Any individual (solution) with a fitness value different than ∞ satisfies constraints 1 and produces binary values for $y_{ijk}, \forall i, \forall j, \forall k$.*

Proof. Two cases can explain a violation of constraint 1. The first case is:

$$\sum_{j \in J} y_{ijk} > p_{ik}, \forall i \in I, \forall k \in K,$$

The value of y_{ijk} is given by the flow in the link between u_{ik} and r_{jk}. However, this flow is bounded by the flow prior to u_{ik}. Since there is only one link toward u_{ik}, and since the flow on this link is bounded by 1, the flow going out from u_{ik} can be either 0 or $1, \forall i \in I, \forall k \in K$. Therefore y_{ik} is binary and cannot be greater than p_{ik}.

The second case is:

$$\sum_{j \in J} y_{ijk} < p_{ik}, \forall i \in I, \forall k \in K.$$

In this case, u_{ik} is necessarily 0, which means that the flow beyond u_{ik} is null too. However, the flow is expected to be maximum with $f(E)$ equal to $\sum_{i \in I} \sum_{k \in K} p_{ik}$. This can be achieved if and only if the flow going from every $u_{ik}, \forall i \in I, \forall k \in K$ is equal to one. This contradicts. □

Lemma 3 *Any individual (solution) with a fitness value different than ∞ satisfies constraints 2.*

Proof. The satisfaction of constraint 2 is guaranteed by the construction of the flow graph. The violation of the constraint indicates that at least one unit of flow arrives at server node without passing any repo-video vertex. However, it is impossible since there is no direct link between user-video and repository vertices. □

Lemma 4 *Any individual (solution) with a fitness value different than ∞ satisfies constraints 4*

Proof. The left side of constraint 4 is the overall incoming flow at server node j. Since the flow going from each repository vertex j to the virtual sink is bounded by the bandwidth capacity b_j, the total flow in the cut prior to repo-video cannot be greater than b_j. □

3) Operations on individuals: With the above algorithms, we are able to encode individuals (solutions) and to evaluate their feasibility as well as their performances. We thus have the basic elements of a genetic algorithm. Now, we present how to generate individuals. Firstly, we show our method to produce one offspring from two individuals. Secondly, we describe our approach for the initial generation of individuals.

The three basic operations for offspring generation are selection, crossover, and mutation. For these operations, the individual is treated per repository. In other words, we extract from the individual encoding of each parent the storage of each repository.

1) The *selection* consists in ensuring that videos that are stored in both parents still appear in the offspring. Such selection process should not be automatic, though. Commonly in GA, the selection process is executed under probability, with regard to a given threshold.
2) The *crossover* operation consists in selecting one video for any free storage spot after the selection operation. Here, videos are randomly picked from the union of the videos stored by both parents.
3) The *mutation* adds some randomness to the process. After the crossover operation, each video has a very low probability (for example $p_{mut} = 0.001$) to mutate to another video that does not exist on the server.

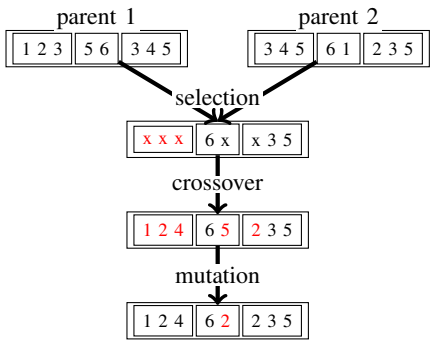


Fig. 3. Creation of offspring: an example with three repositories, having respectively 3, 2 and 3 storage capacities

We give an example in Figure 3. We take two valid individuals in inputs. The selection is done for repositories 2 and 3, but the random pick makes that repository 1 should be generated from scratch (although video 3 is shared by both individuals and would have been kept). For the crossover, the free spots has filled with videos that are randomly picked in the union of sets of both individuals. Finally, the mutation produces that one video is mutated into another, although none of both parents stored it on this repository.

Lemma 5 *The individual produced by the operations from several individuals verifies Lemma 1.*

Proof. In each operation that produces or changes an individual, we prevent duplication of one video to several slots in the same repository. In particular, in the selection operation, we pick videos that are shared among individuals, and in the crossover operation, we select videos in the union of all videos from parents. □

The efficiency of a genetic algorithm depends also on *initial generation*, which should be at least extensible to the complete search space, otherwise, the GA may end without finding any feasible solution. Moreover, an initial generation close to the optimal solution can greatly save multiple iterations of the GA. Our method to endow the initial generation with good quality comprises two steps. Firstly, we randomly choose l videos in each individual to represent the l distinct videos. Then, the number of replicas of a video is decided by the total storage capacity of servers and the popularity of the video. Specifically, the number of replicas for a given video $k \in K$ is set to:

$$\max(\left\lceil (\sum_{j \in J} s_j - l) \cdot g(k) \right\rceil, (m-1)),$$

where $g()$ is the probability density function of the video popularity distribution. Then, we determine the location for each replica so that two replicas of one video should never exist on the same server.

4) Overall description: We implement the elitist **generation revolution** (replacement) strategy to implement our GA. We first create an initial population of v individuals. Then, we produce v offsprings from this population. Out of the $2 \times v$ individuals (v parents and v offsprings), we select the v individuals that have the best fitness functions. If no offspring is produced in $threshold$ consecutive iterations, the algorithm terminates, otherwise it reiterates this process. The best individual is eventually chosen. We give the pseudocode of our GA in Algorithm 1.

Theorem 1 *Any optimal solution of the MCMF instance with a fitness value less than ∞ is also a sub-optimal solution for k-PCFLP with fixed x_{jk} values.*

Proof. From lemma 1, 2 and 5, we conclude that any solution with a fitness value less than ∞ satisfies all constraints in k-PCFLP and binary attribute of two variables. Therefore, it is at least a feasible solution. Moreover, the only cost introduced in MCMF is the transmission cost from user-video node to repo-video node. Since the unit of flow from u_{ik} to r_{jk} corresponds to $y_{ijk} = 1$, the total cost of the MCMF is $\sum_I \sum_J \sum_K e_{ij} p_{ik} y_{ijk}$, which is exactly the unfixed part of the objective function. Thus, the optimal solution of MCMF is also the sub-optimal solution for k-PCFLP. □

Algorithm 1 Genetic Algorithm for k-PCFLP

```
generate initial population G_0
continue = true; t = 0; counter = 0
while continue and counter ≤ threshold do
    continue = false; t = t + 1
    counter = counter + 1
    newPopulation= G_{t-1}
    maxFitness= max{fitness(e) : ∀e ∈ G_t}
    for 1 to v do
        offspring ← Mut(Crossover(Select(G_t)))
        if fitness(offspring) <maxFitness then
            continue = true
            counter = 0
        add offspring to newPopulation
    sort newPopulation according to fitness
    G_t = the v first individuals in sorted newPopulation
return the first individual in G_t
```

C. Parallelizing GA by MapReduce

We are interested in using MapReduce (MR) because of the huge search space and large data set yielded by our k-PCFLP instance. Readers can refer to [22] for a tutorial on parallel genetic algorithm (PGA). We implemented the *dynamic demes* model in our parallelization since it fits better the structure of MR. In this model, the whole population is treated as a single collection of individuals during the evolution. After a new generation is produced in each iteration, the first task of the PGA is to dynamically reorganize demes, which matches the mapping phase in MR. Other operators are independently applied on each deme by reducers. At the beginning of each iteration, the mapper randomly regroups the entire population into r subpopulations (demes), where r is also the number of reducers in the MR system. Each reducer takes care of v/r individuals, and executes GA operators independently. Each reducer produces also v/r offspring, so that v offspring are produced by the whole system. When an offspring is produced, its fitness score is immediately calculated. Only qualified offspring are output by reducers. When all the reducers finish producing offspring, and there is no output, the algorithm stops. Note that the evaluation of the initial population cannot be integrated into the main loop of the algorithm, we illustrate the MR algorithm for the two parts in Algorithm 2 and 3.

1) MR tackles Initial Population: The individuals in the initial population are pre-generated outside of the MR algorithm, so Algorithm 2 does not contain the selection, crossover and mutation operators. The objective of this MR is to distribute evaluation tasks and find the global minimum fitness score.

We assume that the initial population is stored by several chunks in *Hadoop File System* (HDFS). The input of the map

Algorithm 2 MR algorithm evaluates the initial population and finds the global minimum fitness value

```
1  class Mapper: Mapper1
2      method MAP(id, G'_id)
3          for all individual ∈ G'_id do
4              EMIT(rv, individual)
5
6  class REDUCER: REDUCER1
7      method REDUCE(rv, individual)
8          LocalMin ← ∞
9          for all individual ∈ [individual] do
10             fit ← Evaluate(individual)
11             individual ← (individual,fit)
12             if fit < min then
13                 if (fit < LocalMin) then
14                     LocalMin ← fit
15         EMIT(1, (LocalMin, [individual,fit]))
16
17 class Mapper: Mapper2
18     method MAP(1, [LocalMin], [individual,fit])
19         for all LocalMin ∈ [LocalMin] do
20             EMIT(1, LocalMin)
21
22 class REDUCER: REDUCER2
23     method REDUCE(1, [LocalMin])
24         GlobalMin ← ∞
25         for all LocalMin ∈ [LocalMin] do
26             if LocalMin < GlobalMin then
27                 GlobalMin ← LocalMin
28         EMIT(1, GlobalMin)
```

Algorithm 3 MR algorithm produces offspring and finds the global minimum fitness value

```
1  class Mapper: Mapper1
2      method MAP id, G'_t_id
3          for all individual ∈ G'_t_id do
4              EMIT(rv, (individual, fit))
5
6  class REDUCER: REDUCER1
7      method REDUCE(rv, [individual, fit])
8          LocalMin ← GlobalMin
9          for 1 to ⌈v/r⌉ do
10 offspring ← Mut(Crossover(Select([individual, fit])))
11             if (fit ←Evaluate(offspring))< min then
12                 offspring ← (offspring,fit)
13                 if (fit < LocalMin) then
14                     LocalMin ← fit
15         EMIT(1, (LocalMin, [offspring,fit]))
16
17 class Mapper: Mapper2
18     method MAP(1, [LocalMin], [individual,fit])
19         for all LocalMin ∈ [LocalMin] do
20             EMIT(1, LocalMin)
21
22 class REDUCER: REDUCER2
23     method REDUCE(1, [LocalMin])
24         GlobalMin ← ∞
25         for all LocalMin ∈ [LocalMin] do
26             if LocalMin < GlobalMin then
27                 GlobalMin ← LocalMin
28         EMIT(1, GlobalMin)
```

function is the chunk id and the subpopulation G'_{id} stored in the chunk. Then, the map function extracts individuals from the subpopulation. Each individual is attached by the first mapper a random number rv whose value takes from 1 to $r-1$. According to rv, individuals are assigned to different reducers.

The task of the first reducer is to evaluate the fitness value of each individual and report the local minimum fitness in its subpopulation. Particularly, each reducer computes concurrently the fitness value of every individual. When the fitness value of an individual is obtained, it is attached at the end of each individual and compared with a local minimum fitness. If the obtained fitness value is less than the local minimum, the value of the local minimum is updated. After all the individuals are processed, each reducer outputs the local minimum fitness and the set of individuals with their fitness scores in HDFS. Each part of the output data is further divided into two parts: local minimum and individuals. The two parts are stored in two different files. The former is the input of the second phase of MR, which finds the global minimum fitness score.

In the second phase, local minimum fitness of each subpopulation is gathered by the mapper. All the local minimum values are forwarded to a single reducer to calculate the global minimum fitness value. The global minimum is then regarded as the criteria for qualifying offspring.

2) MR produces offspring: The aim of Algorithm 3 is to produce, evaluate offspring, and update the global minimum fitness score. The input of the algorithm consists of individuals in the current generation G_t with their fitness values, and the global minimum fitness. Again, the first map function is used to regroup the subpopulations. The objective of the reorganization is not only to distribute the reduce task but also to exchange individuals in demes and avoid the convergence at a local minimum point.

The main function of the algorithm is undertaken by the first reduce phase. It is responsible for generating and qualifying offspring, and determining the local minimum fitness score. All the qualified offspring and their fitness values are written to the HDFS system with the local minimum after $\lceil v/r \rceil$ offsprings are produced. Then, the local minimum fitnesses are sent to the second MR phase to determine the global one.

3) Complete PGA: Both MR algorithms 2 and 3 successfully transform the centralized GA 1 into its complete parallelized version as shown in Algorithm 4, where the *replace* function is independent of the two MR sub-algorithms. It substitutes the worst individuals in the population with the qualified offspring in a centralized manner.

In the algorithm, map-reduce is mainly used to parallelize the evaluation process of generation. The benefit depends on the number of partitions (number of reducers in our case). Assuming r reducers are implemented, then the evaluation

Algorithm 4 Parallelized Genetic Algorithm for k-PCFLP

```
1  t = 0
2  GlobalMin = ∞
3  Q = ∅
4  Initialize(G_t)
5  min ← Algorithm2(G_t)
6  min ← Algorithm3(G_t, GlobalMin)
7  while Q ≠ ∅ do
8      Replace(G_t,Q)
9      t = t + 1
10     Q = ∅
11     GlobalMin ← Algorithm3(G_t, GlobalMin)
```

process in the map-reduce implementation is r times faster than it is in the centralized implementation.

VI. Evaluation

The inner goal of this Section is to demonstrate that our GA enables large-scale evaluation of real-world cases of telco-CDN serving a lage population of users. Since the management of both a telco-CDN and an ISP network requires tuning various specific parameters, we do not produce a comprehensive evaluation of both push and caching strategies in telco-CDNs. We rather focus on one specific case: the Orange network with a possible deployment of a telco-CDN and a typical VoD service. Through this example, we show that caching strategy is not necessarily the best implementation for the management of telco-CDN since such strategy is blind to the underlying infrastructure.

A. Settings

1) Traces from a national VoD Service: We utilized traces from the Orange VoD service, which is a typical national VoD service offered to the clients of the ISP Orange. The measurement period ranges from June 5th, 2011 to June 19th 2011. We picked 728,931 download requests for 21,385 distinct video from 22,305 unique end-users. More interestingly, we were able to associate each end-users to one of the 13 *regions* that correspond to a metropolitan network in ISP's network. We were unable to track the session duration, so we assume that session duration follows a uniform distribution from 60 to 120 minutes.

We show the service usage on the seven last days in Figure 5. We observe the typical daily pattern, which is common to all localized entertainment service. Two daily peaks happen at around 2:00pm and 11:00pm, and the low activity is around 4:00am. Moreover, we also observe the difference between week days and weekends. These characteristics matches our expectations that our traces are representative of a local, popular service in real world.

We separated these traces into two *periods*. The *warm-up* period represents the input of the prediction of the end-user preference. It also allows to fill the cache in the caching strategy. This period lasts for the seven first days of the traces. The *test* period starts immediately after the warm-up period and lasts x days, for $x \in \{1, 3, 5, 7\}$.

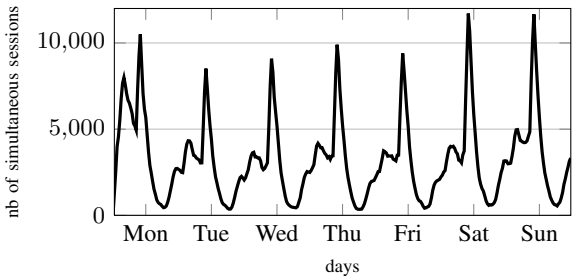


Fig. 4. Service usage

2) Network topology and management: We use the network topology illustrated in Figure 1, except that we consider only one PoP since we are considering only one VoD service. That is, the only PoP is located at Paris. We consider that 13 repositories have been deployed by the ISP so that each router connecting a metropolitan area to the backbone is equipped with a repository. All the telco-CDN repositories are homogeneous and possess the storage capacity of 5,000 videos and out link bandwidth for streaming 1,000 sessions simultaneously.

As for the management of this network, we consider a policy, which is today representative of the management decision that have to be taken by an ISP. There exist three categories of links. The costs are indicated in a unit of money (euros in our case) per kilometer and per transmitted video. Please note that these costs correspond to the context of one ISP [15] at the time we performed this study, but our model is generic so other computations can be done with other context.

The **peering links** connect each repository to the PoP. The traffic on this link goes through the PoP, so it generates peering costs as well as possible monetary compensations to the CDN. We suppose the cost of peering link is 1,000. These are red plain lines in Figure 1.

The **internal links** connect a pair of geographically adjacent repositories in the backbone. They are dashed blue lines in Figure 1. These links are intra-domain links, which are much cheaper than peering links. Therefore, the cost of internal link is set to 1.

The **low-priority internal links** are regular internal links but the network operator experience troubles on these links (they are over-used, or subject to faults). In our simulations, we chose three links: from Lille to Metz, from Lyon to Nice and from Limoges to Poitiers. Since the network operator prefers to use them in low priority, we set a cost of 1,000 but these links are shorter than peering links, so they still represent an opportunity to avoid going to the PoP.

The telco-CDN system operates as follows: an end-user's request is firstly directed to its regional repository. If the regional repository does not have the required video or its bandwidth capacity is over, this repository explores its *cooperation group*. A cooperation group is defined for each repository j as a set of repositories such that the distance to j is smaller than the distance from j to the PoP. Any repository in the group having the requested video and free bandwidth can serve the client. If the requested video is stored within the cooperation group, the

request is redirected to the matching repository. Otherwise, it is a *miss* and the request is forwarded to the PoP.

3) Evaluated Strategies: We evaluated our PGA-based solution to two basic push strategies.

Random placement extracts the videos requested during the warm-up. Then each unit of storage space is randomly filled with one video.

Proportional placement distributes the replicas of each video according to its popularity. The number of replicas is proportional to its request rate during the warm-up, and each repository holds only one replica of a video.

The performances of our strategies depend on the quality of the prediction of users' requests. To investigate the impact of the recommendation accuracy, we produce the prediction of users' preference by mixing up warm-up and test part. Specifically, we replace a certain percentage of records in the test part with records from the warm-up part to generate predictions with various qualities. In particular:

- A **Perfect Optimal** (or Perf-Opt) takes in input of the computation for he k-PCFLP all the requests that will actually be requested during the test period. It is like the recommendation system was prophetic.
- A **Realistic Optimal** (or Real-Opt) takes in input of the computation only requests from the warm-up period. The recommendation engine does not predict anything but just rely on the past.

Perf-Opt and Real-Opt can be regarded as the upper and lower bound of our push strategy.

LRU caching strategy implements the traditional and widely adopted caching strategy. To avoid unfairness due to initially empty caches, we measure from the first request of the test period. When a request cannot be fulfilled at a cache, the repository looks for possible hit in its cooperation group. However, in case of miss, the video is not re-forwarded to these repositories.

B. Evaluations

We enter now in the evaluations of the strategies in our toy-telco-CDN. The main criteria that matters in our study is the overall cost. For each placement strategy, we compare the cost to the one without repositories, that is, we compute the *normalized overall cost* as the ratio of the overall cost from fetching the video to the cost using only peering links.

1) Computation time of tracker: We do not aim here to analyze the performances of our genetic algorithm, and to explore the benefits of our parallelized implementation. However, we can report some basic computation time to express that the implementation of an efficient tracker is possible using today's technologies.

The algorithm has been implemented on a MR cluster consisting of 10 machines with dual 2.70 GHz Pentium processor and 4 GB RAM (however these machines were used by background processes). We set the population of each generation to 500 individuals and the initial population is stored in files with an overall size about 150 MB. For all instances, the algorithm converged in about 350 generations, which took less than 12 hours. With regard to the relatively sub-optimality of our implementation and to the small number of computers involved in this computation, it is clear that the implementation of an efficient tracker can easily be implemented at a large-scale, typically in the context of such service where placement is re-done on a daily basis. Previous works have shown that such daily video re-placement is enough with regard to popularity changes [15]. Please note that the evaluation process is by far the most time-consuming process, so our future works will aim at improving this part of the algorithm.

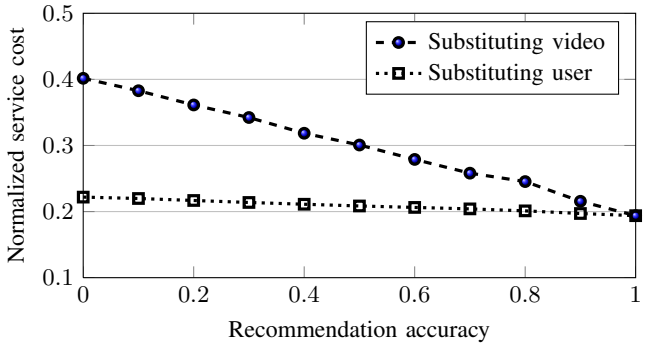


Fig. 5. Impact of prediction on Perf-Opt

2) Impact of the recommendation accuracy: Our tool enables the evaluation of the impact of various parameters on the overall network performances in a fair manner. Here we explore as an example the importance of the accuracy of the recommendation engine.

The recommendation engine provides a set of p_{ik} that reflects whether end-user i will request video k in the near future. We did not implement any recommendation engine, but we *emulated* a recommendation engine with various levels of performance. The recommendation engine outputs a matrix of binary $p_{ik}, \forall i \in I, \forall k \in K$. The input is the set of requests that have been observed during the warm-up period. To generate our lower and upper cases, we did as follows:

- For *Perf-Opt*, the accuracy of the recommandation engine is perfect, so every request of the evaluation period has been successfully predicted by the recommandation engine.
- For *Real-Opt*, the recommendation engine takes the requests from the warm-up and predicts that exactly the same requests will be emitted during the evaluation period.

Between these two extrema, we define a parameter, called *recommendation accuracy*, and noted $a \in [0, 1]$, which represents the ratio of requests that are successfully predicted. In our implementation, it means that $(1 - a)$ predictions are replaced by requests observed during the warm-up. So, Perf-Opt is for $a = 1$, although Real-opt is for $a = 0$. When we replace a recommendation, we can substitute either the requested video or the requester. We show in Figure 5 the performances for a series of accuracies.

Substituting videos gives a larger bad effect on the efficiency of our optimal placement. The influence of the mistakes on users may be canceled by the integration of users' requests at

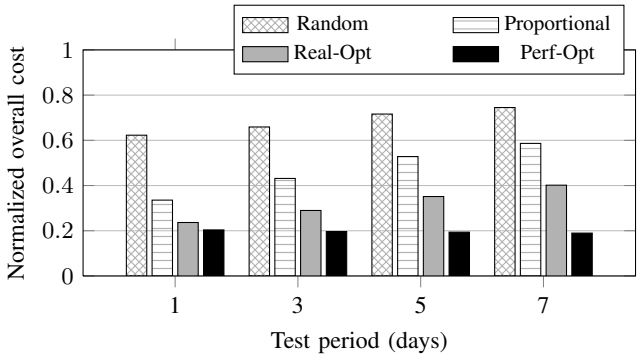


Fig. 6. Normalized overall cost of push strategies

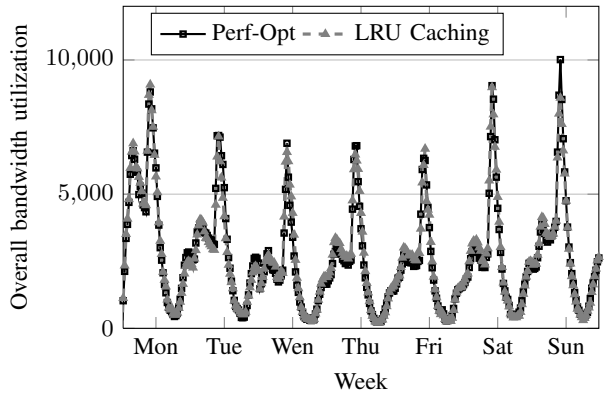


Fig. 8. telco-CDN bandwidth utilization

the regional repository. Therefore, only marginal increment of cost is yielded. However, the error of pushing wrong videos is not compensable, which doubles the overall cost. Thus, we emphasize here the importance of addressing the videos that will be popular in the near future at a regional scope rather than at individual ones.

3) Performances of push strategies: We study the push strategies, especially both random and proportional to the quasi-optimal strategies. Figure 6 shows the prominent performances of our quasi-optimal computation. Although the proportional random strategy is widely accepted as a decent heuristic for placement in terms of hit ratio, we show here that the performances are worse than the optimal placement in terms of telco-CDN management.

When the test period is longer, the performances degrades as some videos that were not requested during the warm-up enter in the catalog, thus are requested. Obviously, the Perf-Opt strategy is not affected by such novel videos, but we observe a significant degradation of the performances of Real-Opt (almost twice less efficient). We mitigate this observation by arguing that placing videos on a weekly basis is not serious when it is possible to do it on a daily basis.

4) Push strategies versus caching: What follows is the main outcome of this paper: we compare, on a given network configuration and a given VoD service, the advantages of a push strategy versus a caching one. The overall performances of LRU caching in term of cost is compared with push strategies in Figure 7(a). Please note that the performances of both caching and Perf-Opt are not affected by the duration time, the former because it is prophetic, the latter because it dynamically uploads the content in the repositories. On the contrary, performances of Real-Opt degrades when no new computation is done.

For the test period of one day, which is the most probable to implement, the performances of push strategies are much better than caching. They achieve an overall cost that is 1.8 times smaller than LRU caching. This result demonstrates that LRU caching is a blind strategy that performs remarkably well for hit-ratio, with almost no implementation cost. However caching is far from ideal for an ISP that wants to actually manage its network as well as its telco-CDN.

The reason of the predominance of Perf-Opt is revealed in Figure 7(b) and Figure 7(c). Both figures count the total number of sessions passed through each telco-CDN internal link. In Figure 7(b), it is visible that a telco-CDN implementing LRU caching uses less internal links than implementing Perf-Opt. In the meantime, Figure 7(c), Perf-Opt utilizes in low priority the low-priority links, which is exactly the desire of the ISP, although LRU caching utilizes these links as regular links (note that they are still cheaper than peering links). In other words, while Perf-Opt allows to engineer the traffic (here to avoid using low priority links but more accurate policies can obviously be implemented), a cooperative LRU caching acts naively.

We represent in Figure 8 the overall bandwidth utilization on telco-CDN repositories. As could be expected, LRU caching performs similarly to Perf-Opt. A naive scientist looking only at such results would definitely adopt LRU caching since it is far easier to maintain, but the results given in Figure 8 emphasizes the opposite. It is remarkable that Perf-Opt performs as well as LRU caching although its impact on the network infrastructure is 1.8 times less severe.

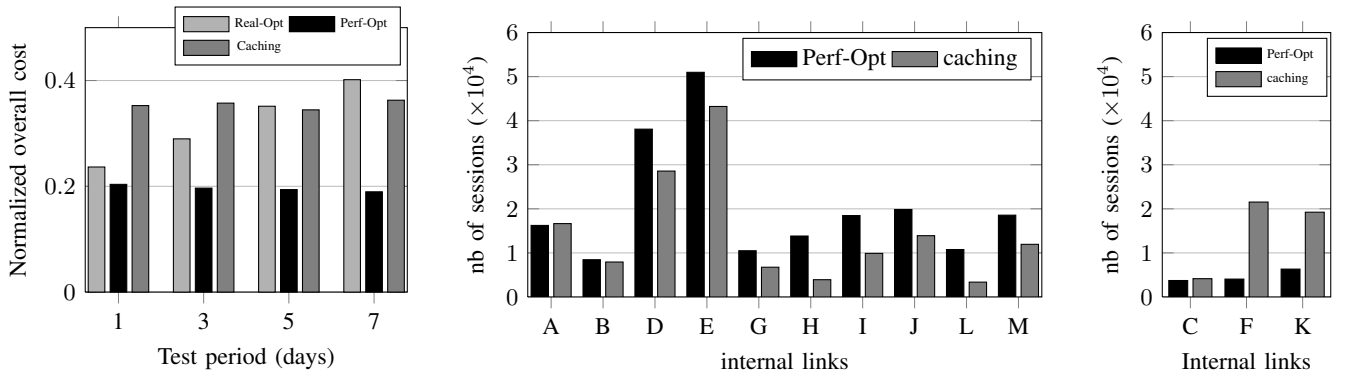
VII. Conclusion

This paper focuses on the design of Telco-CDN, which is a inevitable trend in the development of future network operators. Since ISPs are interested in both hit-ratio and traffic management, we re-open the debate about strategies for pushing video content to repositories. Our new facts are twofold: first, pushing content in an quasi-optimal manner is practically feasible. Second, it makes sense since it enables smart traffic management. We thus open a new perspective of development for research in the area of telco-CDNs. In our future work, we will evaluate our algorithm in a more realistic network topology. Moreover, the evaluation process in the PGA algorithm is quite consuming, thus it needs further parallelization and optimization. Developing the online version of the current algorithm is also interesting.

References

[1] Decision relating to practices concerning reciprocal interconnection services in the area of internet connectivity. Decision 12-D-18, Sep. 2012. French Competition Authority.

[2] Netflix Open Connect Peering Guidelines, 2012. https://signup.netflix.com/openconnect.



(a) Overall cost of Optimal and LRU

(b) Internal link utilization

(c) Low priority link utilization

Fig. 7. Optimal placement versus LRU Caching

[3] Orange and akamai form content delivery strategic alliance. Akamai Press Release, Nov. 2012. http://www.akamai.com/html/about/press/releases/2012/press_112012_1.html.

[4] G. Bertrand, E. Stephan, T. Burbridge, P. Eardley, K. Ma, and G. Watson. Use cases for content delivery network interconnection. IETF draft, Feb. 2013. draft-ietf-cdni-use-cases-10.

[5] B. V. Cherkassky and A. V. Goldberg. On implementing push-relabel method for the maximum flow problem. *Integer Programming and Combinatorial Optimization*, 920:157–171, 1995.

[6] K. Cho, H. Jung, M. Lee, D. Ko, T. Kwon, and Y. Choi. How can an ISP merge with a CDN? *IEEE Communications Magazine*, 49(10):156–162, 2011.

[7] Özgür Ulusoy. Research issues in peer-to-peer data management. In *Proc. of Int Symp. on Computer and Information Sciences*, 2007.

[8] A. Dhamdhere and C. Dovrolis. Twelve years in the evolution of the internet ecosystem. *IEEE/ACM Trans. Netw.*, 19(5):1420–1433, 2011.

[9] J. L. Di-Wei Huang. Scaling populations of a genetic algorithm for job shop scheduling problems using mapreduce. In *2010 IEEE Second International Conference on Cloud Computing Technology and Science (CloudCom)*, 2010.

[10] D. DiPalantino and R. Johari. Traffic engineering vs. content distribution: A game theoretic perspective. In *IEEE INFOCOM*, 2009.

[11] P. Faratin, D. D. Clark, S. Bauer, W. Lehr, P. W. Gilmore, and A. Berger. Complexity of internet interconnections. *Communications & Strategies*, (72):51, 2008.

[12] S. M. Fulton. Vomcast on xbox: Does a cdn, by nature, threaten the internet? http://www.readwriteweb.com/archives/comcast_on_xbox_does_a_cdn_by_nature_threaten_the.php, March 2012.

[13] S. Goel and R. Buyya. Data replication strategies in wide area distributed systems. *Enterprise Service Computing: From Concept to Deployment*, pages 211–241, 2006.

[14] D. Golding. The real story behind comcast-level 3 battle. GigaOM, Dec. 2010. http://gigaom.com/2010/12/01/comcast-level-3-battle/.

[15] F. Guillemin, T. Houdoin, and S. Moteau. Statistics of youtube traffic in orange ip networks. Orange Labs draft, 2012.

[16] Y. He, G. Shen, Y. Xiong, and L. Guan. Optimal prefetching scheme in p2p vod applications with guided seeks. *IEEE Transactions on Multimedia*, 11(1):138–151, 2009.

[17] A. Inc. State of the internet. Technical report, Akamai, Oct. 2012.

[18] K. Jain, M. Mahdian, and A. Saberi. A new greedy approach for facility location problems. In *ACM Symposium on Theory of Computing (STOC)*, 2002.

[19] W. Jiang, R. Zhang-Shen, J. Rexford, and M. Chiang. Cooperative content distribution and traffic engineering in an isp network. In *ACM SIGMETRICS*, 2009.

[20] C. Jin, C. Vecchiola, and R. Buyya. Mrpga: An extension of mapreduce for parallelizing genetic algorithms. In *IEEE eScience*, 2008.

[21] Y. Jin, Y. Wen, G. Shi, G. Wang, and A. Vasilakos. Codaas: An experimental cloud-centric content delivery platform for user-generated contents. In *International Conference on Computing, Networking and Communications (ICNC)*, 2012.

[22] D. S. Knysh and V. M. Kureichik. Parallel genetic algorithms: a survey and problem state of the art. *Journal of Computer and Systems Sciences Internationl*, 49(4):579–589, 2010.

[23] L. I. Kontothanasis. Content delivery considerations for web video. http://www.mmsys.org/?q=node/64, 2012.

[24] D. K. Krishnappa, S. Khemmarat, L. Gao, and M. Zink. On the feasibility of prefetching and caching for online tv services: A measurement study on hulu. *ACM Passive and Active Measurement*, 2011.

[25] S. Lattanzi, B. Moseley, and S. Suri. Filtering: A method for solving graph problems in mapreduce. In *ACM SPAA*, 2011.

[26] J. Leblet, Z. Li, G. Simon, and Y. Di. Optimal network locality in distributed virtualized data-centers. *Computer Communications*, 34(16):1968–1979, 2011.

[27] Z. Li, M. K. Sbai, Y. Hadjadj-Aoul, A. Gravey, D. Alliez, J. Garnier, G. Madec, G. Simon, and K. Singh. Network friendly video distribution. In *IEEE/IFIP Conf. NoF*, 2012.

[28] A. Lodhi, A. Dhamdhere, and C. Dovrolis. Peering strategy adoption by transit providers in the internet: a game theoretic approach? *SIGMETRICS Performance Evaluation Review*, 40(2):38–41, 2012.

[29] Q. Lv, P. Cao, E. Cohen, K. Li, and S. Shenker. Search and replication in unstructured peer-to-peer networks. In *Proc. Int. Conf. of Supercomputing*, 2002.

[30] R. T. B. Ma, D. M. Chiu, J. C. S. Lui, V. Misra, and D. Rubenstein. On cooperative settlement between content, transit, and eyeball internet service providers. *IEEE/ACM Trans. Netw.*, 19(3):802–815, 2011.

[31] M. Mitchell. *An introduction to genetic algorithms*. The MIT Press, 1996.

[32] G. Pallis and A. Vakali. Insight and perspectives for content delivery networks. *Communications of the ACM*, 49(1):101–106, 2006.

[33] G. Pallis and A. Vakali. Insight and perspectives for content delivery networks. *Communications of the ACM*, 49(1):101–106, 2006.

[34] A.-M. K. Pathan and R. Buyya. A taxonomy and survey of content delivery networks. Technical report, Univ. of Melbourne, 2007.

[35] D. Rayburn. More isps not letting cdn place servers inside their network, doing it themselves. http://is.gd/UxExv6, April 2009.

[36] D. Rayburn. Three new white papers released on the telco cdn space. http://bit.ly/GUDrUZ, 2011.

[37] A. Sharma, A. Venkataramani, and R. Sitaraman. Distributing content simplifies isp traffic engineering. http://arxiv.org/abs/1209.5715, 2012.

[38] Z. Shen, J. Luo, R. Zimmermann, and A. Vasilakos. Peer-to-peer media streaming: Insights and new developments. *Proceedings of the IEEE*, 99(12):2089–2109, 2011.

[39] M. Shibuya, Y. Hei, and T. Ogishi. Evaluation for cost-effective cache deployment in isp networks. In *Proc. of IEEE Communications Quality and Reliability (CQR)*, pages 1–7, 2012.

[40] D. B. Shmoys, E. Tardos, and K. I. Aardal. Approximation algorithms for facility location problems. In *ACM STOC*, 1997.

[41] K. Stamos, G. Pallis, A. Vakali, and M. D. Dikaiakos. Evaluating the utility of content delivery networks. In *UPGRADE-CN*, pages 11–20, 2009.

[42] B. Tan and L. Massoulié. Optimal content placement for peer-to-peer video-on-demand systems. In *IEEE INFOCOM*, 2011.

[43] Verizon. Hbo for fios customers. http://bit.ly/JQ2dn8, Jan. 2012.

[44] J. Wu and B. Li. Keep cache replacement simple in peer-assisted vod systems. In *IEEE INFOCOM*, 2009.

[45] W. Wu and J. C. Lui. Exploring the optimal replication strategy in p2p-vod systems: Characterization and evaluation. In *INFOCOM*, 2011.

[46] H. Xie, Y. R. Yang, A. Krishnamurthy, Y. Liu, and A. Silberschatz. P4p: Provider portal for applications. In *ACM SIGCOMM*, 2008.
[47] R. Zhou, S. Khemmarat, and L. Gao. The impact of youtube recommendation system on video views. In *Proc. of ACM IMC*, 2010.